Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Elena Oat

Integrating payment solutions to online marketplaces

Master's Thesis
Espoo, May 30, 2016

Supervisor: Professor Tuomas Aura
Advisor: Katja Ratamäki, D.Sc. (Tech.)

Aalto University
School of Science

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author:	Elena Oat		
Title:	Integrating payment solutions to online marketplaces		
Date:	May 30, 2016	**Pages:**	67
Major:	Mobile Computing—Services and Security	**Code:**	T-110
Supervisor:	Professor Tuomas Aura		
Advisor:	Katja Ratamäki, D.Sc. (Tech.)		

E-commerce and online purchases represent the future of commerce. With this trend, businesses are opening or extending their presence online to gain more visibility, to sell their products and services through diverse electronic marketplaces or online shops, and to engage with their current and future customers to improve the customer experience. This has resulted in an evident need for easy to deploy, reliable and safe solutions to enable an existing web site with selling and payment capabilities. Moreover, people who advertise products or services on their personal web sites but are unable to create their own online stores have realized that they can benefit from the opportunities provided by the Internet by selling their products and services through third-party electronic marketplaces. This thesis focuses on the payment solutions provided by the third-party services, which enable web sites and applications with payment capabilities. These solutions provide a simpler and faster method of integrating a payment system into a web service compared to traditional solutions. Specifically, the research aims at outlining the general principles that must be followed when choosing a payment solution. Integration of payment solutions, implementation challenges and their possible solutions are discussed in a separate section as well. This work is centered, especially, on the marketplace solutions versus the common subscription based services solutions, and one time payments. In addition, this thesis provides an overall review on the latest technologies and business aspects related to the topic. To facilitate this research and to emphasize its practicality, a case study is presented about the integration of a third-party payment method to an online marketplace.

Keywords:	payment, online service, credit cards, electronic commerce
Language:	English

Acknowledgements

Thank you to my supervisor, Professor Tuomas Aura, who has been so helpful both in technical aspects, but more importantly, in encouragement and support on the journey of writing this thesis. I would also like to address an honest and big thank you to Katja Ratamäki along with whom we have developed the Miils service. Finally, a huge thank you to my family who has supported me in this process.

Espoo, May 30, 2016

Elena Oat

Abbreviations and Acronyms

URL	Uniform Resource Locator
PCI	Payment Card Industry
PAN	Primary Account Number
PSP	Payment Service Provider
HTTPS	HTTP over TLS
NFC	Near Field Communication
ISO	Independent Sales Organization
B2C	Business-to-Consumer
B2B	Business-to-Business
API	Application Programming Interface
AJAX	Asynchronous Javascript and XML
PCI	Payment Card Industry
PCI DSS	Payment Card Industry Data Security Standard
EAI	Enterprise Application Integration
SOA	Service-Oriented Architecture
REST	Representational State Transfer
ROA	Resource-Oriented Architecture
MVC	Model-View-Controller
MVT	Model-View-Template
WSGI	Web Server Gateway Interface
JSON	JavaScript Object Notation
DOM	Document Object Model
PaaS	Platform as a Service

Contents

Chapter 1

Introduction

Online commerce, or e-commerce, in its narrowest definition, represents the trading of goods and services on the Internet. Some sources consider also intranets and extranets as media for e-commerce transactions. A broader definition might include social interactions, e-learning, and other collaboration activities that are occurring over the digital networks.

The indisputable growth worldwide of e-commerce is reflected in numbers. In the U.S., it is growing annually 16-17 percent faster than the commerce domain itself [61]. Moreover, the volumes from the first quarter of 2015 have surpassed the indicators from the same period a year before by almost 15 percent [5]. In parallel, European market is observing similar trends. It is estimated that by the end of 2018 retail sales will double in comparison to 2012, according to Ecommerce Europe [14]. Based on the same source, 2014 was registered as a successful year in Business-to-Consumer (B2C) e-commerce. The sales increased during this period by more than 14 percent in total. These figures prove that e-commerce presents tangible opportunities for the new and already existing companies to develop and grow their business considerably if they decide to engage in selling products and services online.

The impact of digital sales are noticeable not merely in statistical data, but also in practical real-life examples. For instance, the coffee house chain Starbucks was able to transform its declining sales into profits due to its digitalization strategy. The enterprise leveraged the social space (e.g., it has worked to become one of the most popular businesses on Facebook with over 36 million followers) to grow its existing and potential customers engagement, and discovered new streams of income by opening online stores and eGift card programs.

E-commerce has experienced a number of changes during the last ten years. Changes that happened during this period touched upon both business and technological aspects. Credit card utilization has been growing, al-

though, according to the World Economic Forum, this trend will not persist in the future. Concurrently, the introduction of Bitcoin has marked another significant turning point in the evolution of online payments, despite the fact that e-cash payments have previously not seen major success [50]. At the same time, with the arrival of Web 2.0, there appeared the concept of social commerce. Social commerce, essentially, implies that consumers possess a powerful influence on the products introduced into the market, as well as on the product development process. The success that is being attained by the emerging players in the Financial Technology (FinTech) domain represents a direct proof of the changes presently occurring in the payments ecosystem [35]. To conclude, the payment ecosystem is transforming together with the technology world.

While online shopping continues to carve its way to the leading positions in commerce, established companies realize that their business models must change. Thus, large enterprises, such as Starbucks, but also middle-sized and small companies (that account for more than one-half of the world's GDP [20]) and especially startups switch their attention to online presence and on exposing their products in the World Wide Web. On the other hand, businesses that have not yet realized these opportunities, or are discounting them, are losing in profits already and will lose in the future, or worse, may cease to exist.

Behavioral habits of customers are switching towards shopping online as well – together with the maturing technology trends in the domain. Online shopping has advanced in its user-friendliness and safety of transactions of funds, and provides an increasingly convenient method of purchasing goods. Consumers are attracted by the possibility of buying products and services without crossing their doorstep, avoiding driving, queues and unreasonably spent time. Besides being beneficial from the economical and business points of view, shopping online provides a number of other advantages too. For instance, users can customize their products in accordance with their preferences even before they are manufactured, and they can also compare products based on price and quality from the different suppliers. Moreover, higher visibility and competition among the providers result in lower prices for consumers [38]. In fact, the possibilities of online shopping increase the likelihood that a user will purchase the service or goods in the first place – it is convenient, quick and often provides more options compared to the traditional shopping.

With all that, a challenge that middle-sized and small businesses often face relates to the lack of finances. As a result, it is often challenging for them to build a solid payment system from the ground up. At the same time, the companies realize the impact of the growing consumer interest in

online retail, and understand that ignoring it will eventually result in loss. That is what has contributed to the appearance of payment solutions on the market. Payment service providers focus primarily on enabling existing services provided by other companies with payment capabilities. In this thesis, the term of payment service providers refers to the companies that provide fully featured payment solutions, which are integrable into existing or separately developed services. A merchant that takes advantage of such services to implement online payments has usually no need to interact with other financial institutions such as banks or payment gateways.

Not that long ago, the negotiations with the payment gateways providers and the creation of a merchant account were the prerequisites for establishing a web store. The fully featured solutions, on the other hand, are faster to implement and are focused solely on providing payment services. These allow new online businesses to start selling within minutes from the registration moment – thus removing the headache of long negotiations with the financial institutions and other involved parties, as well as the laborious paperwork often required in such procedures. This raises a question of whether such solutions are equally trustworthy and reliable as their predecessors. This question presents a reasonable concern – the domain has not matured fully, and, what is more important, is not thoroughly regulated. Therefore, companies that decide to deploy such solutions must be careful about choosing their providers in order to avoid unpleasant consequences. The industry is still in its development phase and not every detail has been considered and reviewed with the same rigor as traditional financial services. However, these services represent an immense benefit to startups and other early stage companies. In addition, since these companies focus mainly on payments solutions, they are able to build more efficient systems than if each of their customers would have implemented one alone. As a side note, these third-party providers are often referred to as Cashier-as-a-Service (CaaS) or Payment-as-a-Service (PaaS), or just payment service providers (PSP).

One benefit that the emerging solutions provide is on-site experience: the users do not have to leave the merchant web store to pay for the services – instead, the payment user interface is integrated seamlessly into the merchant web pages themselves. This proves beneficial both to the merchant as well as to the consumer in terms of the user experience. In addition, online payments with the emerging services are advantageous from the point of view of efficiency (transactions are faster and cost less than using traditional financial institutions) and flexibility [28]. Although, no matter how advanced and powerful some of the payment solutions become, conventional banks will remain a foundational part of the industry, because they are the ultimate sources and destinations of most transactions – customers still need

a bank account to purchase online [11]. The rise of the payment providers is noticeable in their growing number and also in the amounts of investment poured into this market area. For instance, investments in FinTech sector have tripled in the US in 2014 and the same tendency has been registered in the rest of the world [1]. The growing competition among the market players has fortunately resulted in higher maturity and quality of products, which have also become easier to integrate. This in turn benefits the online commerce and, consequently, end-customers. The list of well-known payment providers includes PayPal, Amazon Payments, AliPay, but there exist also smaller and local players.

1.1 Motivation for this work

Physical stores have realized the potential of e-commerce [43] and are extending their presence online. Furthermore, many new businesses have decided to eliminate their physical shops altogether and opened online stores instead.

Over 500 000 businesses open each month in the U.S. alone [6]. A large part of them need to put in place a payment system to start selling through their web store. Naturally, the faster the payment system is launched, the faster the business can start selling its products. Companies selling services and immaterial products can equally benefit from the online presence, if not more than those selling physical products. After all, search engines are the most utilized resource for researching existing products and services. In fact, according to [13], search engines are globally more trusted than traditional media. Moreover, with today's commerce trends, people expect to be able to purchase almost anything with a click of a button. Having the online presence, but more importantly an online store, places businesses one large step ahead of those who do not have one, for obvious reasons.

1.2 Problem statement

Choosing a payment solution for an online service often poses challenges. Similarly does the integration of such a solution into an existing system. The former challenges are associated with choosing the most suitable payment solution from the financial, business, and technical perspectives. At this moment, the number of alternatives on the market only grows, and companies should, at minimum, go through a checklist to decide on their choice. Regarding the latter challenge, the integration of third-party services has become simpler due to the high competition in the sector and the entrance of

new players into the market. Therefore, this thesis presents a tentative set of criteria that a company must consider when deciding on a payment provider. This thesis also focuses on technical and business challenges that a company is likely to encounter when integrating a payment solution into an existing service, as well as their possible solutions. A real-life case study of a company that needed a payment solution for its existing online service is presented as an example and to motivate the work. The solution was implemented as a part of the thesis project.

1.3 Thesis structure

The thesis is divided into six chapters. The Introduction highlights the purpose of this work, while Chapter 2 focuses on the general aspects of the payment solutions, such as their structure and processess. Chapter 3 provides an overview of the business aspects that need to be considered when choosing a payment solution, while the concept of application integration is introduced in Chapter 4. The case study is presented in Chapter 5. Finally, in Chapter 6, we summarize the key points as well as the open questions.

Chapter 2

Payment solutions: their structure and processes

This thesis focuses on the challenges related to planning and implementing a payment solution for a web application. In addition, we investigate the processes required for keeping such a system operational. The following section describes the structure of payment systems as well as their processes.

In order to start accepting credit card payments, a web service needs a merchant account and a payment gateway. This requires time and effort for research and later for the implementation itself. At the current moment, however, the market offers businesses a shortcut solution, which removes the burden of creating a merchant account and a payment gateway. The solution is offered by a payment service provider (PSP), also called a full-stack payment platform, a payment card processor, a technology vendor, or a merchant service provider. In common language, it is also referred to as all-in-one payment solution. It allows businesses to start receiving payments from their customers within hours or minutes.

This thesis focuses, in particular, on the solutions provided by the payment service providers. They are a natural choice for small businesses and startups that need to start selling their products and services immediately and do not require many customized features from the payment system. As a side note, the term startup is referenced a few times in this thesis. To define it, startup is a company that tries to solve an existing problem, which sometimes does not have an obvious solution and its success is not guaranteed [18]. A startup usually either gets traction and grows extremely quickly in terms of employees and its customers, or it disappears. Large number of startups are initially funded by their founders, who later seek additional funding from Venture Capitalists (VC).

The list of well-known global payment service providers includes Brain-

Tree, Stripe, Amazon Payments, PayPal. There are also other services, which target more focused geographical areas such as Dwolla and Trustly. Dwolla operates only in the U.S. market, while the Swedish company Trustly provides its services for a range of countries in Europe. Additionally, there exist other smaller providers that target their own region and, sometimes, their neighboring countries.

When choosing a payment service provider, each business is faced with a choice of which payment solution to implement and maintain. Since the market in this area offers hundreds of them, the task becomes time-consuming. Of course, not each service is available everywhere — some solutions can be deployed in particular geographical regions only.

2.1 Participants of a credit card transaction

It may not be obvious to a purchaser, but there are several parties which participate in the processing of an online payment transaction with a credit or debit card. In this work, it is considered that the payment is performed with a credit card for simplicity and for consistency with the service which represents the main subject of this research. In addition, it is assumed that a payment service provider was chosen for payment processing, which offers a larger range of services than those provided by a regular payment processor. In this case, a payment performed with a credit card involves five parties [41], [33]:

1. Merchant: the entity who is selling the products or services.

2. Customer: the entity who is purchasing the products sold by the merchant with a credit card.

3. Issuer: usually a bank which has provided the credit card to the customer. The issuer has access to the client's account and manages the transfer of funds on the client's behalf. Hence, the issuer transfers the money for the purchased product from the customer to the merchant's bank account.

4. Acquirer: merchant's bank, which has opened the account for the merchant. Its tasks also include managing the transfer of funds on behalf of the merchant. Hence, this entity receives the money on a successful purchase and deposit it to the merchant's bank account. These are also called processors and provide sometimes in addition to their core activity same services as the next entity in the chain does – the payment

service provider. However, the usual case involves a processor and a reseller of its services, which is the payment service provider.

5. Payment service provider (PSP): the middle party which connects financial institutions, i.e. banks, with the online services that sell products. Direct interaction between banks and web portals does not usually happen, due to security reasons. The advantages of utilizing a PSP instead of a processor include value-added services that the PSPs provide. In addition, through payment service providers companies can access multiple processors for whom the PSP resells services. Moreover, not all acquirers, in fact, provide direct connection to their networks. Consequently, a third party is necessary.

These parties are presented in Figure 2.1, together with the communication flow that happens between them.

2.2 Message flow of an online payment

The communication between the parties mentioned above happens in the following order:

1. The customer selects the product to purchase from the merchant's website or a marketplace and enters the credit card details for the payment.

2. The website dispatches in the back-end an authorization request to the payment system provider, which in turn addresses this request to the issuer.

3. The issuer responds with a positive or negative response, depending on whether the authentication of the customer has been successful and whether there is a positive balance or sufficient credit for the purchase. Consequently, the payment system provider delivers this response both to the website back-end system and to the acquirer.

4. Finally, the issuer transfers the money for the purchase to the acquirer. The issuer then charges the customer's account, while the acquirer transfers the money to the merchant's bank account. It is worth noting that the merchant does not acquire the whole sum of money paid by the customer but has to pay the fees for the PSP services.

5. In case the purchase succeeded, the acquirer sends a payment receipt to the merchant's website through the PSP, informing about the successful money transfer.

Below is a flow chart of the described communication process:

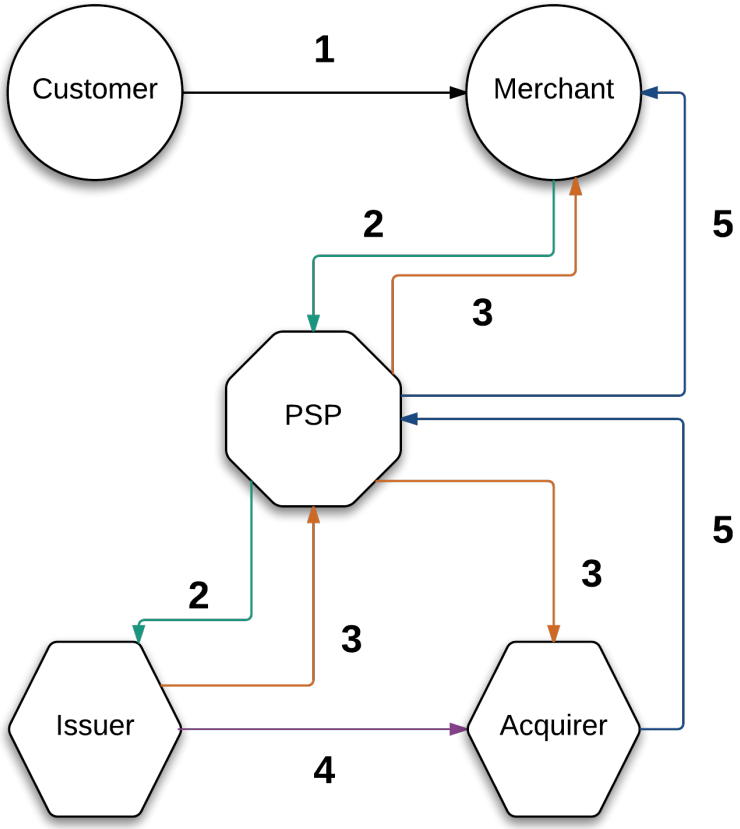

Figure 2.1: Communication flow of an online payment

2.3 "Full-stack" solutions

At the moment, small businesses are provided with an option of choosing a "full-stack" payment solution, provided by the PSPs, in which the payment service provider plays a greater role and takes care of many of the processes that usually are the merchant's responsibility. This simplifies and speeds up the initial configuration process, as businesses do not have to apply for a merchant account (basically, the clients are utilizing PSP's merchant account) and a payment gateway. In case of Stripe [57], the setup process consists of filling an online form with required details about the business (not all business types are permitted, however). Consequently, the business can start

selling within minutes.

Delaney [10] presents a list of third-party payment solutions that small companies might resort to in order to start selling fast. For instance, the Dwolla [12] payment solution is mentioned in the same list, which provides a solution for companies selling products with a relatively low price. On the other hand, an all-in-one solution might not represent an optimal choice in the long run, as it is more costly when a large amount of transactions are processed on the daily basis. In such cases, it is worthwhile for the business to open its own merchant account at the acquirer in order to save on the per-transaction charges.

Chapter 3

Payment solutions: business aspects

This section focuses on the questions that a company must consider from the business and operational points of view before implementing a PSP solution. We also review the participants that are involved in the payment processes as well as their contribution and compensation.

3.1 Future of payment service providers

In the near future, payment service providers will start supplying value-added services in addition to their currently offered services [8]. From a business perspective, PSP customers will receive updated information about transactions directly into their management systems. End-customers, on the other hand, will benefit from the customizability offered to them through PSP services: more targeted suggestions and promotions, calculated based on previously purchased products. The latter trend has also been noticed by the World Economic Forum (WEF), which also suggests that this trend will raise the end-customer engagement in product development [19].

Another direction observed by WEF is related to the virtualization of money. Bitcoin and other electronic currencies may cause a disruption in the traditional financial industry. They are particularly advantageous from a cross-border value exchange perspective.

The seamless experience that retail customers face when purchasing products and services (e.g., paying for Uber within the app) is, likewise, becoming a trend. The key drivers for these changes are both the development in hardware technology (e.g., wearables, NFC) and the rise of mobile applications that allowed companies originating from the IT industry to enter the payment

ecosystem.

The prices for services provided by payment system providers may be reduced in the future by linking the bank accounts directly into the payment system, without intermediaries such as credit card associations. In fact, this practice is already implemented by a number of PSPs. Simultaneously, the transaction costs incurred by credit card associations will gradually disappear, with the decrease of the credit card and, especially, cash utilization. Concurrently, the differentiation of brands and design will fade as the payment process will become seamless for the customers [19]. This phenomenon is emerging partly due to the appearance of digital wallets, produced by the major technology vendors (e.g., Google Wallet, Apple Pay). A wallet is simply a software component that consumers install on their mobile devices. The software is then linked with the bank accounts or may contain data about the credit cards, loyalty cards, and gift cards. Consequently, the owners of digital wallets are not required to present their plastic cards to the cashier in a store, but they may utilize their digital wallets for a quick checkout [32].

Independent Sales Organizations (ISO), as merchant service providers are sometimes called, are struggling to differentiate their niche, as many are focusing on diverse areas being "any payment solutions" to "any merchants". However, they realize the importance of developing one core area of expertise. For example, Stripe, one of the leading ISOs, works hard on simplifying the adoption of payments into existing online services. Their Stripe Relay product embodies a useful but simple idea of selling products through third-party apps. Another solution provided by the same company was created in cooperation with the cab service Lyft, for which Stripe built a solution that allows drivers to receive payments for their rides almost instantly, compared to previous delay of several days.

3.2 The payment method

As it was mentioned in the previous section, the currency as well as the preferred method of payment are among the most important factors to be considered when choosing a payment service provider. To open the discussion even further, we would like to underline that the choice of the actual payment method is nonetheless important. Companies must carefully consider all pros and cons when deciding on these. The customer's current preferences regarding the method of payment are probably amongst the most substantial criteria. However, there exist other factors that can influence the choice too.

This topic is examined in detail in [2]. The choice of whether accepting credit cards, Bitcoin or any other payment method depends on a number of

aspects. For instance, credit cards are widely accepted worldwide and are advantageous for customers because of their repayment flexibility. However, their level of adoption differs from market to market. While MasterCard and Visa are extensively used in the USA and some parts of Europe, other countries prefer their own local brands. Furthermore, transactions performed with Visa and MasterCard incur high interchange fees, which impact negatively on merchant profits. Further, this financial burden is reflected in higher prices, which affect the consumer in the end.

Debit card transactions result in processing fees as well. However, the fees incurred are regulated by neutral parties, for example in the U.S. by the Congress. Bitcoin as a form of payment may be favorable from the point of view of lower processing fees imposed on merchants. Further, cross-border payments with Bitcoins cost considerably less than with other methods. A compelling feature to merchants is the irreversibility of Bitcoin transactions. Thus, merchants will not have to worry about payments reversed to payers, as may happen in case of MasterCard and Visa. On the other hand, Bitcoin has detrimental features as well. For example, its exchange rate changes often, the currency remains unstable, and the issues with the maximum global transaction rate remain unsolved. Furthermore, not every consumer is familiar with electronic currencies, such as Bitcoin.

The choice of payment methods depends largely on the business and its priorities. It is, however, fortunate that several payment providers offer a number of these methods, instead of just one. Indeed, one of the attractions of the full-stack solutions is that they can aggregate many payment methods to one service, letting the customer choose but without increasing the complexity for the merchant.

3.3 Selling your own products or being a marketplace

Depending on the business idea, a company can be selling its own products or, instead, offering a service that facilitates sales between other companies and end-users.

In the first case, the ways to implement payments are relatively clear and present a relatively simple implementation case. Besides, most of the payment service providers have a ready solution for this case.

The second case – the marketplace solution – requires a further investigation. Specifically, the business needs to decide whether it will act as a middleman and accept payments for products to its own account and only

after transfer the money to its merchants, or will it rather provide autonomy to the merchants and let them handle most payment processes (such as resolving disputes) on their own. The former choice results in a more resource-consuming project in the long term. However, it provides a set of benefits to the marketplace merchants: in case of any payment-related issue, the marketplace becomes responsible for resolving it, as well as for paying the handling fees. On the other hand, the latter choice is less straining for the marketplace provider, as it leaves all the responsibility for the products to their sellers, including managing refunds and answering to customer questions.

3.4 PSP choice: points of consideration

When choosing the merchant services provider, the most important questions to be asked are:

1. How long is the setup process: is it a few days or a few months before the business can start accepting payments?

2. How hard it is for the developers to integrate the solution into their existing software platform? This phase includes checking the documentation provided by the payment service provider and how up-to-date it is. In addition, it is essential to see which programming languages the libraries for integration are written in. That way the businesses are able to estimate better the implementation time. If using a web framework for building the web service, are there any existing third-party packages that one can use to integrate the payment solution more quickly? In addition, how reliable and trustworthy are they in terms of updates and maintainability, as startups often, if not always, choose an open source solution, which may have limited long-term support.

3. A significant role in choosing the provider is played by the quality of its customer support. Businesses can contact prospective providers directly to investigate their offers. While researching the options, they acquire knowledge of their support level: are they responsive to their potential customer's questions, how professional is their attitude towards their clients, and so on. In addition, it is worthwhile to ask about all the possible fees the business will have to face, about the stumbling blocks businesses usually encounter when using their services, and further details and documentation about the offered services. It is useful to ask about the provider's experience with other companies

and which specific services they have used. Besides the above points, it is worthwhile checking what other businesses implemented that particular payment solution.

4. Does the provider accept credit card payments in all the countries of the target market? Does it allow transactions between the target market countries and the country where respective business is registered?

5. Does the provider offer a suitable type of service? For instance, not every merchant services provider offers a marketplace solution. A marketplace, in terms of this thesis, means a service where the end user can be both a retailer and a purchaser.

6. Payment methods supported by the solution.

7. The costs that the company will have to pay to the provider for its services. When making an analysis of costs for the merchant services, businesses must be aware of hidden costs. Such costs may be fees for failed payments, currency conversion fees and rates, and fees when paying with a particular card type.

8. What is the distribution of risk between the entities participating in the payment process: the payment service provider, the merchant or marketplace, and its customers. For small businesses, especially, it is essential to ensure that the company does not spend additional money and time on solving financial disputes. Thus, if the company provides a marketplace, it could decide to make the end-users selling their services or products responsible, as was already mentioned in subsection 3.3. In that case, the business acts as a middleman between user-merchants and user-customers without being financially or legally responsible for the quality of the product provided by the user-merchants. The marketplace business merely facilitates the process of product and money exchange. This, however, must be stated in the terms of service.

9. Another important aspect that must be considered are the payment preferences of the target market. For example, customer preferences on payment methods differ considerably on the European and U.S. markets. Americans choose plastic card payments (either credit or debit) over any other payment method. According to data provided in [63], payments with a plastic card in the U.S. constituted 71 percent of all online payments in 2012. On the other hand, in Europe the numbers of card payments amounted to only 46 percent [15].

10. Currencies and languages supported by the payment provider [44].

11. The methods of dispute handling by the payment provider. According to [10], the process of disputes settlement by the payment processor represents one of the important factors for selection. The same source mentions that it is equally important to know the conditions of the plan termination.

    Dispute case in context of payments arises when a customer is dissatisfied with a purchased product and requests a refund, but the provider does not agree or is unable to provide the refund, e.g., customer does not inform provider of sufficient reasons for the dissatisfaction, or refund functionality is not enabled on the application. As a result, customer contacts credit card organizations for reversal of the funds. The process of reversing paid money through credit card associations is called chargeback. Such cases are reviewed in further detail in the case study presented in Chapter 5.

12. Ability of performing purchases within the same application, without being redirected to the payment provider's pages.

There exist also comparison services that outline the differences and similarities between diverse payment providers. These may provide additional assistance when deciding on the appropriate service.

3.5 Who gets the slices of the pie?

Each of the participants of the payment process, naturally, needs to be paid for their services. Hence, the money that a user paid for a product is split into many pieces and each of the participants receives their share.

In the case of all-in-one solutions (e.g. Stripe, PayPal), the merchant selling a product must pay a relatively small fee to them [39]. A part of these fees constitute the so-called merchant discount. The discount represents the fee that the acquirer will have to pay to the issuer for processing [31]. The merchant discount also includes the interchange fee — the amount charged from the acquirer to the issuer for its transaction services. Additionally, there's an assessment fee charged by the credit card networks, such as Visa and Master Card.

To summarize, each transaction that is processed by a PSP will have a fixed cost and a variable cost (calculated as a percent of the transaction, called also processing fee). The fixed cost, also called transaction fee, cannot be negotiated and it includes all the fees that the PSP must pay to the issuer,

acquirer, credit card associations and other parties involved. The variable cost is the percent of the transaction sum that is paid to a PSP for its services. This cost can be lowered and negotiated before signing the contract with the PSP and it may depend on the monthly transaction amounts and other related details.

Small and medium-sized businesses prefer a flat-percentage solution offered by PSPs (such as PayPal, Stripe, etc.) that avoids the complexity of the pricing incurred by all of the involved parties. This helps them conveniently estimate beforehand how much they must pay to all of the parties down the payment system chain.

Another fee which includes, in turn, the PSP fee, is the one specified by the marketplace itself. Thus, merchants selling their products on a digital marketplace essentially pay one fee — the one applied by the marketplace, but which includes a whole chain of other fees, which are paid by the marketplace to the PSP and to the other participating parties.

Below is an example of the cost breakdown for a meal plan sold by a registered dietitian on a digital marketplace. This example showcases reward distribution when using Stripe as a PSP. In this particular case, the marketplace charges 30 percent of the customer price. Similarly, Amazon charges on its Kindle selling platform 30, but sometimes 70 percent of the cost of the purchased book. The pricing applied by Stripe is valid for the U.S. and was calculated at the time of writing. The acquirer and issuer bank's applied fees as well as the credit card network's assessment fees represent approximately the actual fees which may vary from case to case.

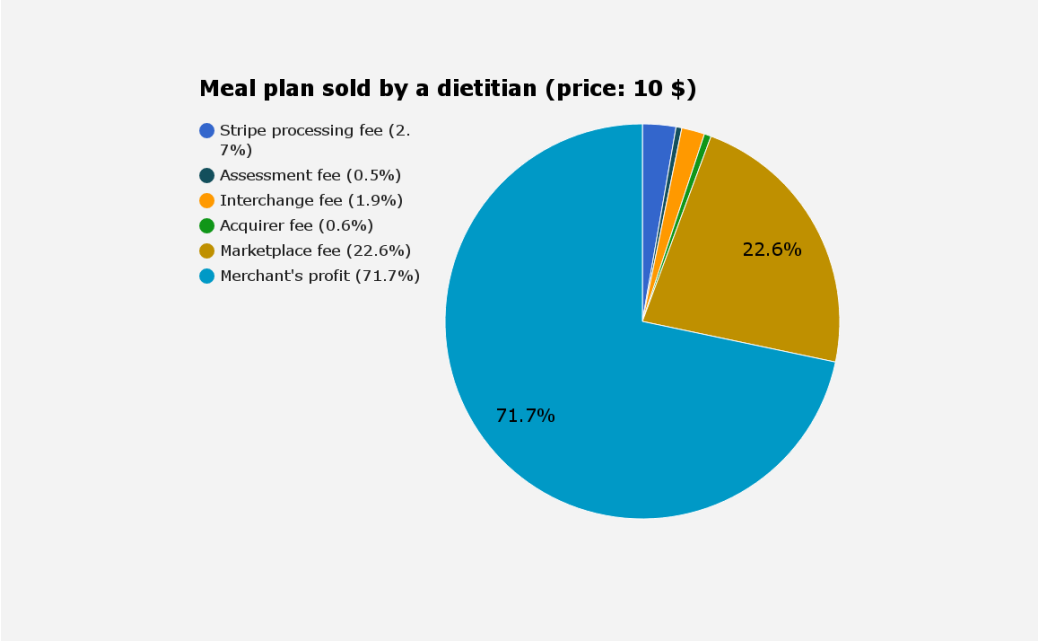

Figure 3.1: Example cost breakdown when utilizing a marketplace as a platform for merchandise

3.6 Why paying to a marketplace?

From the chart shown above it becomes evident that the fee applied by a marketplace for its services constitutes a significant portion of the product price. Therefore, an explanation is needed for why a small business would pay such extra fees to a third party. Consider, for example, a business intending to sell hand-made jewelry to a larger audience than just those that visit their boutique in the city center. They decide to post the product catalog online so that anyone on the Internet would be able to find the products and purchase them. This particular merchant might know nothing about how to create a website and, more importantly, how to integrate a payment solution into it. Even if they have the expertise, the merchant may not want to spend time or energy on it. A natural choice in that case is posting the products on an existing online marketplace where anyone can advertise and sell products. In return, the marketplace charges a fee for its services.

Although a merchant selling on such a platform must pay an amount of money to the marketplace on each transaction, the merchant also benefits from its services in a number of ways [39]:

1. The marketplaces may create new communities of buyers and sellers and, thus, create new markets that did not exist before. Established marketplaces, on the other hand, may control entry to the market that they originally created, which leaves the merchant no choice.

2. No need to deal with paperwork related to compliance with the Payment Card Industry (PCI). PCI defines a strict set of rules, called PCI DSS [46], that all companies that store, process and transmit cardholder data have to follow. In fact, the merchant will not even see the card information of the paying customers. This, consequently, removes the burden of complying with the above mentioned strict list of rules. However, the merchant must still comply with essential security practices, which include securing and authenticating all transmitted data, keeping all software up-to-date, installing firewalls, etc.

3. Trustworthiness of the marketplace adds credibility to the merchant's services. According to [13], 59 percent of people who shop online were more likely to make a purchase if the payment method was provided by a payment service provider with a known brand, such as PayPal or Amazon Payments.

4. Ability to receive payments through different payment methods, supported by the platform: debit and credit cards, PayPal, Bitcoin, etc.

5. Simplified application process for the merchant, in comparison to applying for a merchant account with an acquiring bank.

6. Other benefits that result from using a robust and reliable platform, which has implemented security and fraud protection. A small merchant might not spend enough time or might not be aware of all of the threats when designing their own payment system. Additional factors, such as tested user experience and design, may save a small merchant from the trouble of doing it on their own.

7. The costs for using a marketplace are predictable. In case of PayPal and Stripe, the fees consist of a fixed fee and a percentage of the transaction. The fees are a percentage of the turnover, which removes the risk associated with upfront investment. In case of a merchant account, the fee calculation is more complex (but, nevertheless, well defined).

3.6.1 Relations between main parties of a marketplace

Essentially, the participants of a marketplace can be separated into three groups: the marketplace service provider, the merchants who sell their products through the platform, and the customers, i.e. users who purchase the products available on the marketplace.

Each of these groups has their own rights and obligations, which need to be defined and stated in the service's terms of service.

Another important party — the one who in fact plays an essential role in the payment processes — is the PSP. Both the marketplace and the merchants possess a PSP account (in case the marketplace chose this type of management), and consequently, a legal agreement with it. For instance, in case of Stripe, the agreement between Stripe and the marketplace provider is called Stripe Connect Platform Agreement [60].

There can be distinguished two types of relations: relation with the PSP and relation with the marketplace. The merchants and the marketplace are those who have a direct relation with the PSP. End-users, in turn, as well as merchants, have a direct relation with the marketplace. Customers are indirectly related to the merchants when the marketplace sells its products on the merchants' behalf. These relations are schematically shown below:

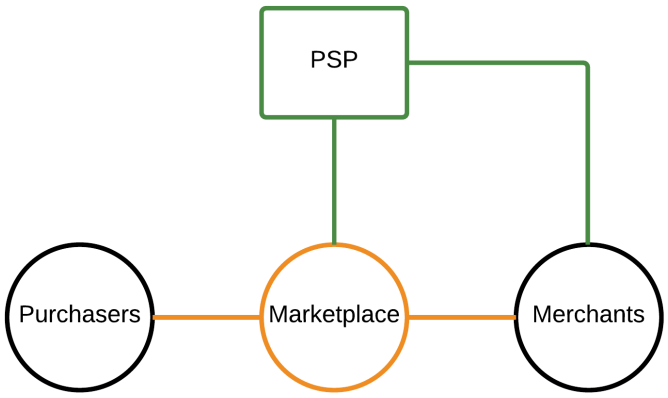

Figure 3.2: Green: relation to the PSP; orange: relation to the marketplace

3.7 Online payments in Finland

Finland is the only Nordic country where payments with credit cards are not the most preferred choice of online payment. Finns choose e-payments, i.e.

online bank transfers with immediate confirmation to the merchant, in most cases when making purchases online. Hence, analysing the market, target customers and their preferences represent an essential step in the PSP decision process. If the payment solution is meant to serve the Finnish market, among the possible PSP offerings is Paytrail Payment Service [45]. It offers a large choice of payment options including e-payment. This payment option is missing from, for example, the Stripe offering. Thus, Paytrail has a solid advantage on the Finnish market of online payments. However, there are also considerable price differences between these two providers. Paytrail charges a monthly fee, besides the one-time payment fee. In addition, the one-time payment charge exceeds the one offered by Stripe. This example shows that the customer preferences may be just as important as the transaction fees when choosing a PSP.

Chapter 4

Application integration

Online services that decided to use a PSP, need to integrate the payment solution into their system. The provider simplifies this task considerably: the business does not need to apply for a merchant account to a payment gateway. Nevertheless, the integration process still requires work. The work load varies depending on the PSP chosen and the other variables mentioned in Chapter 3.

4.1 Application integration

To grow their business and remain competitive on the market, enterprises must constantly search for new cooperation and ways to serve their customers better and provide them with more value. As noted in [4], the number of business-to-business (B2B) cooperations in supply networks will increase because enterprises realize the importance of specializing on their own core and growing their competitive advantages.

In online services, the B2B collaboration process often requires exchange of data and communication between back-end applications of different organizations. This is when the integration question becomes relevant – how to achieve an effective and automatic communication between two different businesses that differ in both their processes and data? Ideally, an integration would require minimum time, resources and effort. Among the goals of successful integration of information systems are removing human interaction and automating the involved communication processes, as well as avoiding major changes to the existing systems.

4.1.1 Definition and challenges

In general, we can differentiate two types of integration tasks: those that are implemented between the systems within an organization, and business-to-business (B2B) application integration. As defined in [7], B2B integration in the context of software technology is the infrastructure that connects back-end application systems within enterprises to its partners through message exchange protocols. One example of such protocols is the electronic interchange (EDI).

There are several challenges involved in integrating information systems, whether they belong to the same enterprise or to different ones. As mentioned previously, the systems are heterogeneous in their functionality and processes as well as the consumed and provided data. Secondly, systems are autonomous — they have internal processes and components that change their state independently of other systems, without interaction with the other cooperating parties. Therefore, the integration layer must be aware of these internal state changes and react accordingly. Lastly, systems are distributed — explicit communication is needed to inform other systems of any changes in their state. Again, integration layer needs to assure the communication between such systems [7].

In [24] another set of challenges is mentioned: scalability, dynamic configuration, semantics, and discovering relevant data. With the addition of more data sources, systems will need to accommodate all of these sources of data. Thus, they must scale rapidly together with the growing needs.

Integration of each new data source is resource and time consuming. Ideally, this integration should happen automatically and provide data availability immediately. In addition, semantics must be introduced to the integration processes, so that interfaces carry a descriptive meaning. With the growth of the data sources, the amounts of data will become increasingly large and it will become challengingly hard to discover relevant data. According to [24], as much as 70 percent of all IT related expenses can go towards software integration in companies.

4.1.2 Evolution of application integration

Application integration is a well-established concept, which has already experienced a number development phases. In its first phase of development, there was point-to-point integration. This type of integration, in its essence, implies that each isolated system is directly connected to all of the other participating systems. Point-to-point integration phase was followed by the enterprise application integration phase (EAI), in which there exists a common

platform that receives messages from different systems, adapts the received data, and afterwards delivers it to its destination.

EAI and other early integration technologies were successful in integrating applications, but they had a number of limitations (e.g., integration is achievable only with nodes within the same LAN). Then again, the Internet allows information retrieval originating from distant sources within seconds. Further, the Internet has learned to deal with the limitations of the firewalls. As a natural consequence, the Internet technologies have started being leveraged in application integration too [27].

Web services as a pattern represent another phase in application integrations. Web services are, essentially, web applications but without human interface because they exchange information primarily with other electronic systems. According to [52], the term web service means a well defined abstraction of computational or physical activities which involve resources and which are targeted at fulfilling customer's needs. Web services are almost always delivered through the HTTP protocol.

A web service also represents a type of application programming interface (API). Through public and private APIs, heterogeneous systems are able to communicate and exchange data, as well as exploit services provided by the other enterprises.

There can be distinguished three architectural types of modern APIs: the service-oriented architecture (SOA), the resource-oriented architecture (ROA) - a RESTful architecture, and the event-driven architecture (EDA) [9].

The SOA architecture is oriented towards services. In a system built on SOA principles, its interfaces represent services. For instance, an interface which translates the text from one language to another represents a service.

To define what a resource-oriented architecture is, it is helpful to introduce the concept of Representational State Transfer (REST). REST is an architectural style which defines a set of restrictions on how a protocol should be designed for it to be called a RESTful protocol. ROA is a RESTful architecture. It is based upon the concept of a resource. As examples of resources serve diverse pieces of information, such as weather forecast for tomorrow, or today's exchange rate. Each of these resources has a unique URI. Further, there are defined five methods for accessing or manipulating these resources: GET, HEAD, PUT, DELETE and POST [34], [36].

In the event-driven architecture an event that occurred (e.g., the start or finalization of a process) is disseminated to all of the parties interested in receiving it [40].

4.2 Integration of payment provider interfaces

To use payment provider solutions, software developers often consume their application programming interfaces. To simplify integration tasks, the providers usually have implemented libraries in the most common programming languages. These libraries serve as the building material for calling the payment provider interfaces. Consequently, the developer tasks consist of integrating these into the existing codebase and then consuming the needed services. As a side note, a few of the providers, such as PayPal, offer both SOA and RESTful interfaces for their payment solutions.

When using web development frameworks, the libraries are often available as ready-to-use apps that could be plugged into an already existing system. This simplifies the integration process even further, considering that the app provides built-in functionalities for the most common payment service needs (e.g., charging for purchases, refunding the purchases).

Integration becomes more complicated, however, when a company encounters a mismatch of company and consumer preferences (called business-to-customer mismatch in [25]). This situation is specific to companies that are facing the need to install multiple payment gateways in order to satisfy their customers needs. As an example, some customers may prefer to pay with PayPal, instead of any another method. For this reason, the companies must carefully consider the payment methods of the marketplace, as has been already noted in subsection 3.4. The marketplaces must ensure the satisfaction among the majority of the target audience.

To summarize, below are presented the most common steps that are involved in the integration process of a payment solution:

1. Creating an account with the PSP, i.e., registering for the service and providing all the required information about the business, as well as choosing the type of the payment system (e.g., marketplace, personal merchant web store).

2. Learning about offered functionalities and their corresponding API calls. Generally, PSPs usually provide documentation, as well as a few test accounts that can be utilized to experiment with these functionalities.

3. Creating a development environment and simulating a real-time payment system. Payment providers, such as Stripe, provide their clients with testing accounts as well as with a list of bank card numbers for diverse cases (e.g., one for the case of a simulated opened dispute, an-

other for the case of a canceled payment). These can be utilized to test the implemented functionalities within the development environment.

4. Switching to the live account and enabling the payment system on company production servers.

Among the existing marketplace solutions in the current market of payment providers are Stripe Connect with its Standalone Accounts option. In this solution the marketplace merely facilitates the selling-purchasing processes, while merchants have their own Stripe accounts. The Stripe Managed Accounts option implies a customized solution, where the marketplace business performs all of the payment system management — an almost opposite of the Standalone Accounts solution where most of the payment processes are managed by Stripe instead of the merchant. Similarly, the BrainTree Marketplace, available in the U.S., provides a somewhat comparable to Stripe Standalone Accounts solution for creating marketplaces. The WePay Online Marketplace offers another analogous solution that is available in U.S. MANGOPAY, on the other hand, offers a marketplace solution to companies operating in Europe.

4.2.1 Common patterns in the payment integration

Below are briefly described the common patterns and functionalities encountered during the payment system integration. These serve as a background information for Chapter 5. In addition, by presenting them here we aim at emphasizing the common aspects that a developer must implement during the payment solution integration:

1. Compliance with the PCI DSS: Compliance with the Payment Card Industry standards becomes a pertinent topic when a business decides to accept payments with credit cards. In its essence, PCI standards are a set of rules that a company handling cardholders' data has to follow in order to avoid the risks of sensitive data compromise. These standards also apply to the companies that benefit from the third-party payment services. However, often the requirements for these companies are not as rigid and the list of requirements is shorter than for those that directly handle, store and transmit cardholders' data.

   Marketplaces and individual merchant web stores must familiarize themselves with the PCI DSS requirements, and assess their belonging to defined categories. If these companies outsource their payment services to payment providers, they usually qualify as A or A-EP, which means that they must satisfy the Self Assessment Questionnaire (SAQ) A or

A-EP eligibility criteria, respectively. The difference between these two categories may seem minor. However, it is significant from the security point of view. In the case of A category the merchants outsource their payment management as well as payment web pages fully to a PCI compliant third party. In the case of A-EP, the page on which a consumer pays for a product is managed by the web store back-end, which then redirects all the necessary information to the third party payment solution. Above all, in neither of these cases the merchants are allowed to store or receive receipts with the cardholders' data in an electronic format. However, the data can be delivered and stored as printed papers [47].

As an example, the Stripe customers using the stripe.js library are eligible for SAQ A, the category which has a slightly less demanding set of rules imposed by the PCI. To achieve this, the payment pages powered by Stripe service contain an inline frame (an example of it is displayed in 5.1) that is loaded from the Stripe domain.

While PCI compliance is generally an essential initiative in ensuring the security of payment procedures, it carries a number of disadvantages too. The compliance is a continuous process that does not stop at a particular moment. Instead, the compliance must be evaluated on a regular basis, together with the product development. This process is especially burdening for small companies that are not able to constantly invest money in these activities. For this reason, PSPs that offer simpler compliance procedures are generally preferred. On the other hand, those companies that disregard the PCI compliance suffer corresponding consequences. The fines that are applied to non-compliant companies range from 5 to 100 thousands U.S. dollars. In worse cases, companies can even lose the ability of performing any operations with the credit cards [51].

2. Webhook notification:

   A webhook is a notification sent by the payment provider's server to notify the merchant about changes related to its customers. These notifications are sent asynchronously. One example of such events is a dispute case opened by a customer. In order to receive such events, the merchant's back-end system has to explicitly subscribe for them.

   Webhooks are provided by the majority of payment vendors. The main advantage of webhooks is their push instead of pull notification characteristic. Besides resource saving, webhooks also deliver immediate

updates about any occurring events. To define in general terms, webhooks are an event exchanging mechanism overlaid over the Web [9], and they exploit the HTTP protocol to deliver messages.

Examples of events to which a marketplace could subscribe include opened disputes for transactions, merchant account changes, and subscriptions. These events often require an immediate action from the marketplace. For instance, in case a user requested a refund, the marketplace's back-end must start the refunding process.

To configure webhooks, developers must provide an endpoint where the notifications will be sent. An endpoint implies a publicly reachable interface. The endpoint must, naturally, be available at all times. The URL for the endpoint must be configured in the payment provider's dashboard (a web page through which the marketplaces and the merchants customize their payment system settings). The marketplace back-end then parses the relevant messages and takes an action corresponding to the event.

3. Authorizing the marketplace: This concept is pertinent to the marketplaces – the online stores that sell products on behalf of the product owners. Merchants, just like the marketplace, acquire a PSP account when they become marketplace retailers. By joining the marketplace, the merchants agree to provide access to their PSP accounts to the marketplace. For authorization, most marketplaces implement the widely known OAuth2 standard. A more detailed description of how marketplaces are authorized to perform sales on behalf of their merchants is specified in Chapter 5.

4. Well-known payment service providers (PayPal, Stripe, BrainTree, etc.) offer to their customers an opportunity of reducing the burden of PCI compliance. This becomes possible if the special pre-formatted payment forms (e.g., Drop-In UI by BrainTree, Stripe Checkout by Stripe) are utilized by the marketplaces on their payment pages.

4.3 Integrating payments to existing services

This section focuses on a concrete real-life example of integrating a payment solution into a web service. As the case study subject serves the Miils web service, described further in Chapter 5. Below are addressed some of the points discussed earlier in this section.

4.3.1 Integrating third party applications into existing web applications

As mentioned earlier, the payment service providers offer software libraries for easier integration. Although the existence of such libraries is helpful, these are not enough for accomplishing a fully functional integration. The libraries merely provide a set of functions for the API calls and other common procedures. To implement a fully featured payment system, applications need somewhat more complex capabilities, such as updating the database, inserting new data as the new events occur, modifying user interface, and so on.

Generally, web frameworks, including Django, support the concept of reusable applications. In essence, any web application can become an independent and reusable package of code. This alleviates the problem of re-writing the same code for exact same purposes over and over by different developers. Thus, the developers utilizing a specific web-framework co-create an entire ecosystem applicable to this framework. The ecosystem provides an extensive list of applications that can be hooked into an existing web app for desired functionalities. All that is achievable with a relatively low effort.

When choosing a third party service, such as a payment service provider, developers must consider the following aspects [37]:

1. What is the ecosystem on which the third-party service was built? For example, if the third-party service is built using Ruby on Rails, but developers building the marketplace utilize Django, it is essential to research whether the payment provider has also knowledgeable Python developers. It may happen so that Python is out of their competence.

2. Existence of other companies exploiting the third-party service and utilizing the same technologies as the future marketplace.

3. Possibility of retrieving all of the data accumulated by the third-party service. In case the third-party service becomes unavailable, their client companies should be able to retrieve and save all the data to their own servers. Additionally, developers must consider the complexity of integrating retrieved data into another third-party service in case of a negative outcome.

4. Tutorials in the relevant programming language of how to utilize the service.

5. Update frequency of the third-party packages provided for the relevant web framework. It may be that the third-party service does not directly

provide first party binding for a particular framework, in which case the company must analyze if there are any web applications written for this third-party service by other developers for the corresponding web framework.

6. Third-party service's activity, and presence of a community in diverse development forums and platforms, such as StackOverflow.

Chapter 5

Case study: Stripe payments integration into an existing service

This chapter focuses on a real case study: a startup company that has existed for two years and had to deploy a payment system for its online services. First, we will describe briefly what is the service and the idea behind it. Further, we will explain the payment solution as well as will reflect on the challenges faced during its design and implementation.

5.1 Miils: a platform for planning suitable meals

Miils is a social platform for people who want to follow more closely their daily nutrition or for those who would like to change their lifestyle. These changes may be triggered by finding out about an existing allergy that a person has, or because a person would like to switch gradually to a healthier lifestyle, in general. Additionally, the platform was thought to be an interactive community, with people sharing their stories and solutions to challenges that they faced related to nutrition.

The service is a web application, accessible on the desktop and mobile via the browser, considering its responsive design. Native application development is planned for the near future, as the customer interest grows.

The focus of this thesis is on payments solution implemented for this web platform. Currently, Miils is aiming at becoming a platform where dietitians and other professionals in nutrition would come to the site to share their diet plans and recipes with their potential customer base. The price for a professionally designed diet plan becomes more affordable as it is shared among multiple customers. Instead, if developed individually, each plan may

be costly and, as a result, is hardly affordable.

Ultimately, if the model works, the platform would become a marketplace where certified professionals act as merchants and regular visitors acts as customers. In addition, anyone would be able to sell a recipe or a meal plan, however, these will not be marked with a professional stamp of "verified by a dietitian".

5.2 Introduction to the technologies and tools used

Below we will present a list of technologies and tools that are commonly used in developing web applications. These definitions will help the reader to better comprehend further details of the implementation, presented in later sections.

5.2.1 Web frameworks

Web frameworks are widely used in the industry of web application development. These provide a large set of benefits: streamline development process that is achieved by automating some of the mechanisms, more structured codebase that results also in more readable code, reuse of components which results in easier, faster and less prone to errors development process, support for parallel development of content and components [48].

Web frameworks represent, in essence, a toolset using which web applications (or web apps) can be developed in a more efficient and rapid way. Besides practical tools and automation of processes, web frameworks provide benefits from the security point of view too. For instance, the development of session management functionality implies a very careful and thorough approach. Fortunately, most web frameworks offer this functionality out of the box. Another example is automatic XSS sanitization provided by web frameworks – instead of implementing it for each case, it is available as a built-in functionality [62].

As a rule, web frameworks support and promote the Model-View-Controller (MVC) architecture for developing applications. The idea behind MVC is separating logical parts of an application from its view, i.e., its visual representation. The model component describes data and the business logic for how to return the data. The view component is responsible for generating a user interface containing the requested data. Finally, the controller monitors user inputs and communicates with the model to request the necessary data.

Django is an open-source web framework, written in Python programming language. As a common trait to other web frameworks, it abstracts the most used web development patterns. This, consequently, saves development time and eliminates code repetition.

The framework also follows the MVC architecture, only in the case of Django it carries a slightly different name of Model-View-Template (MVT). In MVT views serve as the MVC controller component, and templates are responsible for visual representation, while each model represents one database table. Views contain most of the application logic. Often one view is associated with one visual page.

There are two approaches in Django of how to define a view: class-based or function-based. Miils uses function-based approach, i.e., each view of Miils app is a function. To define it, a view in Django is a function which receives as parameters a request object and any other additional parameters. Inside the view the app may request some information from models or update them. A template is usually defined inside the view too, which is subsequently rendered as an html page. The view returns a response object, which includes the template, the request, and any context variables for rendering inside the template.

Django is among the most popular web frameworks to this day. A large list of well-known web services were built using Django, such as Instagram, Pinterest, Mozilla, National Geographic, etc.

Miils web application has been developed using Django web framework.

5.2.2 AJAX

AJAX stands for Asynchronous Javascript and XML. The term was defined first time in 2005 by Jesse James Garret in his article [21]. According to Garret, AJAX consists of a number of technologies, which allow for partial page reloads instead of full page reloads. This represents a common pattern in web development, because often a page needs only sectional updates and very rarely the entire page reload. Therefore, less content travels from the server to the browser and vice versa. This saves bandwidth and enhances user experience by improving page loading times.

AJAX is implemented in applications using Javascript libraries and frameworks. The web page returned to the client contains Javascript code in the form of functions and event listeners. These functions intercept user actions and decide on whether to send a request to the back-end servers or solve the user request on their own (e.g., data validation). In case a request is handed out to the back end, this processes it and returns a response back to the front end. The response is usually returned in JSON or XML format. Subse-

quently, the Javascript functions receiving the response update the relevant parts of the page.

5.2.3 Document Object Model and Javascript libraries

Document Object Model (DOM) is an interface for modifying an HTML page using programming languages (e.g., Javascript). A web page is a document which contains DOM nodes and objects. Using DOM, programs access the nodes and the objects and modify their content. They are also able to add new nodes or remove the existing ones.

Web applications often utilize Javascript libraries for manipulating the DOM. Commonly used for these purposes libraries include jQuery, React.js, AngularJS, jQuery UI. These libraries abstract the prevailing in web development patterns of DOM manipulation, graphical user interface modifications, etc. Third-party providers often provide their own Javascript libraries too, to facilitate the integration process for their own services. For example, Stripe payment provider has implemented Stripe.js [56] and Checkout.js [55] libraries. By utilizing Stripe.js library, merchants simplify the applied to their businesses PCI DSS requirements, considering that credit card numbers are not stored on the merchant servers, but are posted directly to Stripe infrastructure. Stripe's Checkout.js library, in turn, builds on top of Stripe.js, and provides payment forms which can be plugged into the existing development code without any substantial changes. Thus, Checkout.js also reduces the overhead related with the payment system integration. The list below presents the advantages of Checkout.js library utilization in further detail:

1. The library is updated on the periodic basis. The browsers download the library from the Stripe Content Delivery Networks (CDN), so developer intervention for updates becomes unnecessary.

2. The look and feel features of the payment forms are customizable. For example, to add the brand name to the payment forms requires merely one parameter addition.

3. The greatest advantage of the library by far is the fact that the payer credit card details never hit the company database. As a result, PCI DSS compliance requirements are reduced considerably. However, one requirement becomes a must, which is serving all the pages over the HTTPS [54]. Stripe Checkout library inserts an inline frame (or iframe) into the payment pages, which appears as a part of the same page to the users. The credit card details and other sensitive data is inserted within this iframe, which is served off of the Stripe domain. The data

inserted into the iframe is directly posted to the Stripe servers without the interaction with the merchant's back end [58].

Miils application exploits the Checkout.js library due to its benefits. A caption of an iframe loaded by Checkout.js is displayed below:

Figure 5.1: Iframe inserted by Checkout.js

5.3 Miils system architecture

As it was mentioned earlier, Miils is a web application developed with Django web framework. Miils is hosted on Heroku, which is a platform as a service (PaaS) cloud service. As with other cloud services, Heroku reduces the overhead of setting up and managing an application infrastructure. Basically, it removes the need of managing server installations, their updates, up- and down-scaling, and other maintenance tasks that are usually involved in managing the hardware and software components. Among the advantages also are the acquired mobility. Operations team can work on projects from anywhere while being able to add or remove hardware resources or any other required components with a click of a button. Developers, likewise, are able to deploy apps from any place and at any time [26].

Following are the main components of the architectural setup of Miils application, which is hosted and maintained by Heroku and other cloud service add-ons:

1. Django app: Miils web application
2. WSGI server: Gunicorn, the WSGI HTTP server built on Python

3. Database management system (add-on): PostgreSQL, an open-source object-relational database system

4. Search Engine (add-on): Elasticsearch, an open-source highly scalable search engine

5. SMTP provider (add-on): SendGrid, a service using which the application can send emails, newsletters to its customers and users.

6. Application monitoring (add-on): NewRelic, a service that allows to monitor the app performance, page load times and other indicators.

7. Caching (add-on): MemCachier, a distributed cache system.

The above setup is rather typical for a web application hosted on a cloud platform.

Besides the mentioned above components, Miils service uses also a number of APIs and services provided by the third party companies. For instance, besides its own local authentication, Miils has implemented Facebook's Login and Google Sign-In service. These were added for user convenience in the signup process. For static file storage and serving, Miils uses Amazon S3 services. For the marketplace capabilities of hosting both merchants and customers Miils consumes Stripe Connect APIs.

The figure 5.3 below presents Miils architectural parts, showing also some of the third-party services.

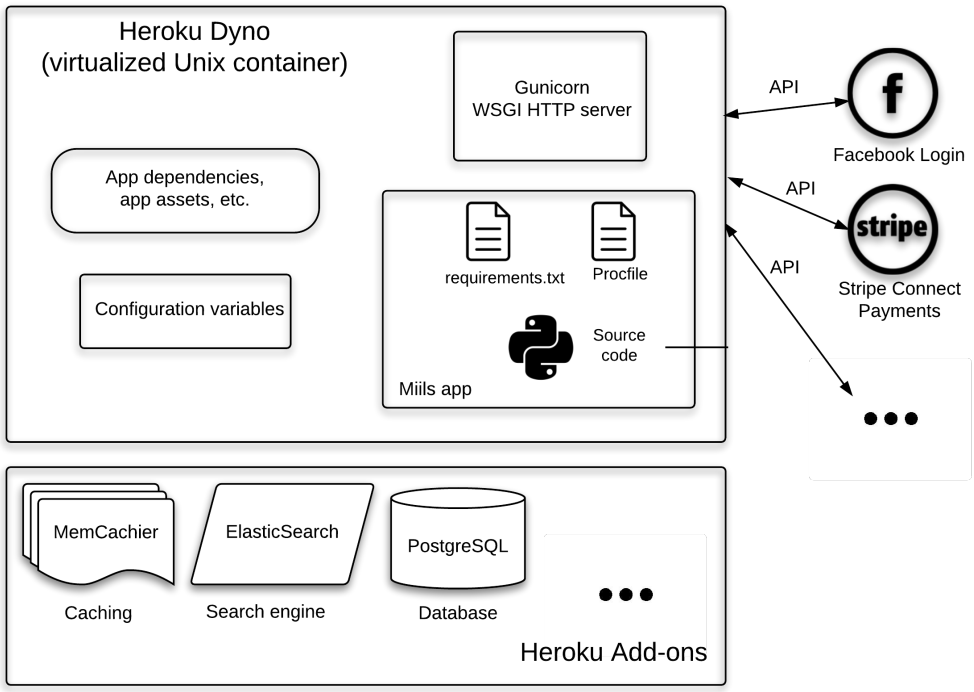

Figure 5.2: Architecture components of the Miils web app

5.4 Payment process in Miils

An authenticated user can purchase any item on the website within seconds thanks to the earlier mentioned Stripe payment forms. Purchases are performed only with a credit card: Master Card, Visa or American Express. To this end, the user is asked to enter all the necessary payment details, which can also be remembered for future purchases. One thing to note is that there is no such thing as shopping cart in Miils. This method of implementation is advantageous because it removes the redirect to another page, and creates a feeling of a single page application. The verification process lasts just a couple of seconds, after which the item purchased becomes available for review. To the user the process appears straightforward and intuitive. However, a number of things happen behind the scenes:

1. When the user clicks on the "Purchase" icon, this triggers a click event. The event is caught by a Javascript function defined on the page. This

function is listening for "click" events initiated by some specific elements. These elements are differentiated from others by their class, which is assigned to all "Purchase" icons. To explain, a class of an HTML element is a global HTML attribute. Using classes Javascript and CSS can select and access elements of the DOM. The information about the clicked element is retrieved inside the body of the mentioned function. From the data attributes of the clicked element are extracted the marketplace Stripe publishable key, the merchant's name, the price of the item and its name.

2. Subsequently, this information is passed to a function defined in Checkout.js library.

3. This Checkout.js function in turn renders a payment form to the customer.

4. The customer enters the credit card information: card number, CVC, expiration date, zip code and an email address, to which will be later sent the receipt. The customer then presses the "Pay" button. The Checkout.js form manages the form validation, e.g., it verifies that all of the required fields have been entered.

5. The Checkout.js library exchanges information with the Stripe servers via HTTPS and verifies with the credit card associations that the credit card is acceptable for payments.

6. As a callback argument of the last information exchange between the Miils back-end and Stripe is received an object, having a number of properties. For our purposes we retrieve merely the token sent by the Stripe servers. The token is a unique ID (28-bytes, "tok" + unique combination of ASCII characters) returned by the Stripe server to initiate the charge. As it is mentioned on the Stripe's page [55], Checkout.js never charges, it only creates tokens that can be used for charging in the back-end.

7. In this same callback function are updated the values of a hidden input field loaded on the payment web page.

8. Finally, the form is submitted and browser sends a POST request to the back-end.

9. In the Django view to which the request is sent, we retrieve the item that is being purchased by querying the database with received information. Subsequently, merchant and buyer are retrieved from the

information passed to the view. Finally, by calling Stripe APIs we charge the buyer on merchant's behalf. All the information about the transaction (buyer's details, merchant's details, time of the transaction, etc.) is saved in the database for further reference.

10. Depending on the success of the operation, we either show a success message and update via AJAX the purchased item or show a transaction failure message and provide details on further steps.

The presentation of the purchased item differs slightly depending on the page where the item is being purchased and the type of the item. A graphical representation of events described above is presented in the figure on the next page:

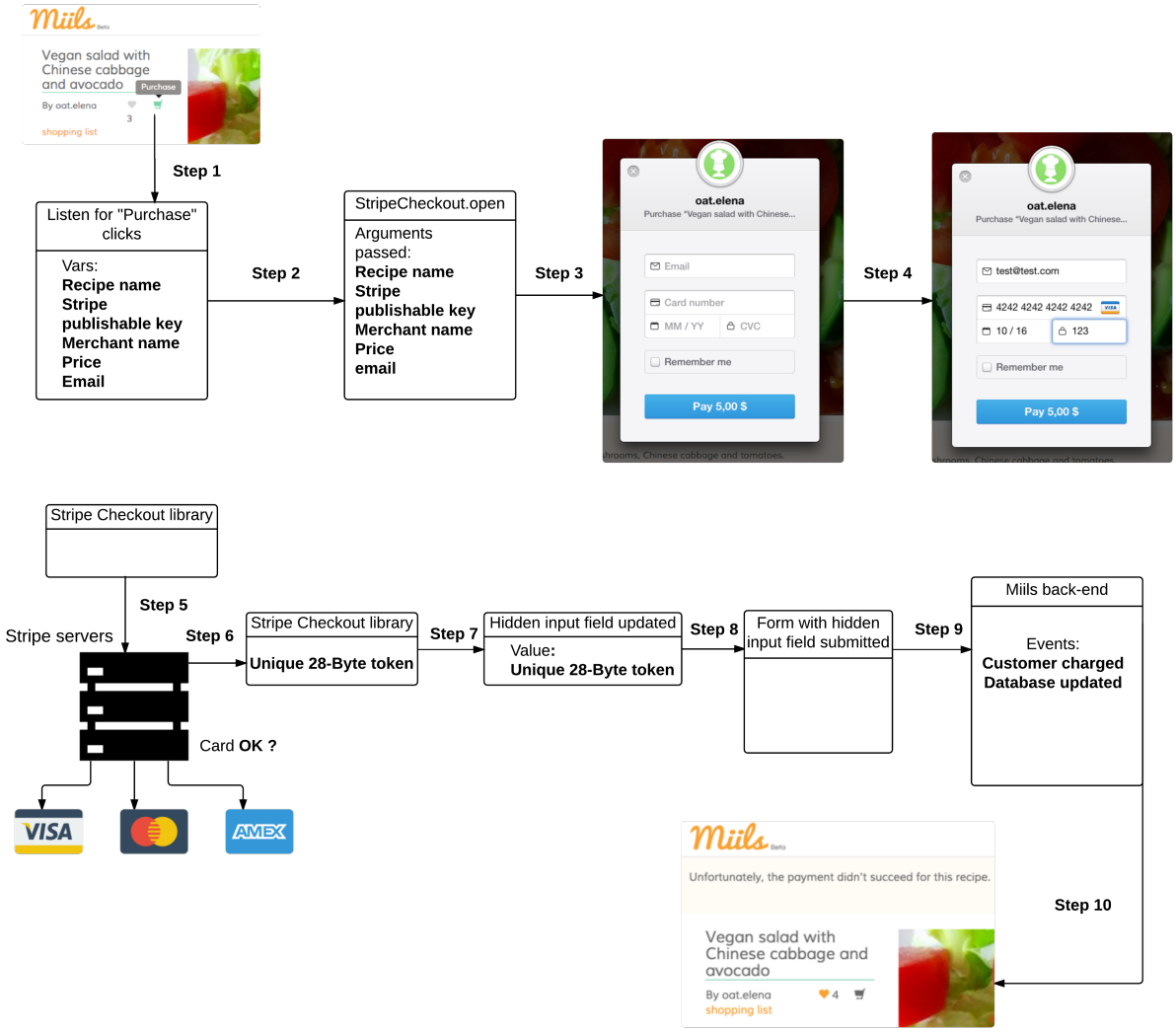

Figure 5.3: Payment process flow

5.5 Becoming a merchant

Any registered user can become a merchant in Miils. The procedure of becoming a merchant is quick and straightforward: the user must press the "Connect to Stripe" hyperlink on the profile page. Subsequently, the user is redirected to the Stripe Connect page, where user's personal and business details must be entered. In case of the U.S., an individual can sell recipes or plans by merely providing EIN number or Tax ID. Naturally, an indi-

vidual owning a company may sell items too, while the registration process differs only slightly. On the Stripe Connect page users review their terms of service and a corresponding agreement with the Stripe provider, called "Connected Account Agreement", before authorizing the Miils application to accept payments on their behalf.

The hyperlink for starting sales in Miils includes a number of query string parameters. These parameters help the Stripe provider to identify the marketplace, on which the merchants will be selling their products. Parameter client_id is a unique number assigned to each Connect Platform. Another one–the redirect URL–represents the page to which the user will be redirected after the authorization process has been completed.

By clicking "Authorize access to this account" on the Stripe's page, the users authorize Miils to accept payments on their behalf, to create customers on their behalf and they allow the access to their data, that is necessary for processing payments for them. If the user has agreed to provide all of these rights to the Miils app, the Stripe servers redirect the user back to the Miils website. There are a few things that happen at this point in the back-end:

1. The Miils back end receives a code from the Stripe to obtain the authorization credentials, which was included as a query parameter in the redirect URI. This code is valid for 5 minutes and that can be used only once. The Miils back end then sends a POST request to an oauth Stripe node with the received code.

2. If the request was successful, response's payload includes the merchant's publishable key and an access token, which is a secret key, assigned by Stripe for this specific merchant. In addition to that, Stripe also sends Stripe user ID for that specific merchant.

   In general, each Stripe Connect Standalone account (for the case of Miils, each Miils merchant account) is assigned two API keys: a public and a secret one. When the Miils marketplace sells on behalf of the other individuals, it requires the credentials to authenticate and authorize its access before Stripe for managing the so-called connected accounts (Miils merchant accounts created with the Stripe Connect).

The flow of a successful authorization is shown in the sequence diagram below:

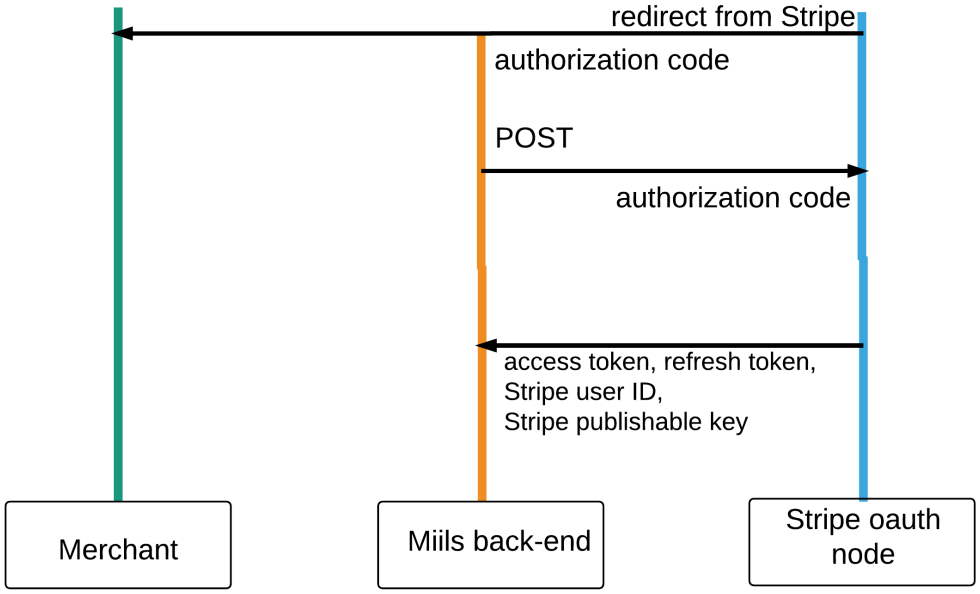

Figure 5.4: Authorization flow: merchants authorize Miils to sell on their behalf

The Miils marketplace is authorized by its merchants for the management of their accounts using OAuth2 framework [29]. This framework was developed to allow a third-party service to access an HTTP service on behalf of an account owner, that was registered on this HTTP service. The framework removes the requirement to share the account owner's credentials with a third-party in order to access the account owner's data stored on the HTTP service. Instead, the third-party receives an access token using which it can perform all the necessary operations against the HTTP service on behalf of the owner.

Stripe provides two levels of permissions that third-party services can obtain once they are authorized: read and write permissions. The first level allows the third-party services read-only access to all of the data related to the charges, refunds, customers and other details [59]. The second type permits the third-party application to modify some of the owner's data. For instance, the third party can refund the purchases on behalf of the account owner.

In the case of Miils, the read and write permissions are required, which means that we can both read the information on sales as well as modify it (e.g., charging merchant's customers, providing refunds) on behalf of the

merchant. With all that, marketplaces created with Stripe Connect cannot access the credit card information of neither of their customers, nor of their merchant's customers. Anyhow, this is advantageous for both the marketplace customers, as well as the marketplace itself: customers are assured that the marketplace cannot misuse their cards, while the marketplace must comply only to the "lighter" version of the PCI DSS.

There are two ways in which the marketplace authenticates and authorizes its access to the merchant's data: using the merchant account's API keys, that are received during the OAuth2 authorization, or using the marketplace's secret key and the merchant's Stripe account ID. According to Stripe [53], the second method is more secure and must be favored, except for some cases. One of such cases is when, a mobile application intents to communicate directly with the Stripe servers, and view or modify data on behalf of the merchant. In this specific case, the marketplace provides the merchant's secret key to the application, so that it can perform requests on merchant's behalf. If the second method were used, the marketplace would have needed to send its own secret key to the application and anyone with the app installed would have been able to access this key and, as a result, view and/or modify the data for the rest of the marketplace merchants.

Through all of its interactions with Stripe, Miils utilizes the second method for authorizing access to the merchant's data. For authorization using OAuth2, Miils exploits rauth Python library [49].

5.6 Challenges faced in the implementation process

This section describes some of the challenges faced in the implementation phase of payment system of the Miils app, which are likely to be faced by the other digital services. To explain these in further detail, a short description of the problem and solution are presented.

5.6.1 Refunds

One of the challenges that was encountered during the project development phase was related to refunds.

Refund is a compensation paid to the payers for a product that they had decided to return back to the merchant. The reasons for returning a product are diverse: from unsuitable size to low satisfaction with the product.

The question of whether to provide refunds to users reflects a delicate

subject. On the one hand, the level of trust that the users experience towards the marketplace is higher when there exists a well-defined user-friendly policy. As a result of it, the sales normally grow. On the other hand, the marketplace and its merchants may lose a part of the profit by allowing refunds. Especially, if there exists no particular requirement for returns. Lastly, the misuse of the refund policy can be prevented only partly. Moreover, in order minimize the misuse the marketplace must act pro-actively and initiate necessary measures before any products are sold.

In the Miils app any user can buy a diet plan, and they can also request a refund for it. Thus, there exist risks of misuse. After all, anyone could register and buy the plan, copy it or just consume the information, and then refund it. Practically, any registered Miils user could pursue this opportunity.

Inevitably, the question of whether to provide the refunds has to be discussed with a lawyer. Regulations related to purchaser protection differ from country to country, and depend on the type of goods that are sold (tangible or intangible) as well as other factors.

However, many startups and small-sized businesses do not possess necessary funds to hire a lawyer. Besides, legal aspects of company founding appear usually as mundane and requiring such scarce resources as focus, time and money [3]. Therefore, startups either disregard these, or often resort to the Internet to find answers to their questions. In better case scenarios, they turn to the student-attorneys for help. As expected, though, a mere online search for the regulations does not produce a sufficient result, as the terms of service are often, if not always, a tailored service.

A non-exhaustive research done by us on the topic of terms of service for digital goods, shows that these differ substantially depending on the business and products offered. Some US services, such as Google Play, allow returns for an app within 2 hours of purchase. For other digital goods, such as books, user gets a much longer period [23]. Other services, such as a Miils competitor, Gatheredtable does not refund any purchases made, according to their terms found at [22]. The service also uses Stripe payments, just as the Miils app does, however, not necessarily in the same format and on the same conditions. In their terms, Gatheredtable mentions that any issues arising from Stripe service are Stripe's responsibility.

Regarding the laws in Europe, a customer can ask for a refund for digital goods, before these are streamed or downloaded. Directive's main points can be found at [16]. Despite the existance of such a regulation, many companies (e.g., Facebook) do not follow this directive in their terms of service [17]. This may be explained by their different origin (e.g., company founded in the U.S.), even though they do provide services in Europe too.

According to a review executed by a professional lawyer regarding the

right of cancellation of digital services provided by Miils, its consumers must be warned before opening the purchased meal plan or recipe of losing their right to refund their services immediately after this act.

In other words, it is insufficient to merely copy the terms of service from a related service. Even slightly different products and services may require different terms of service.

5.6.2 Chargebacks

Another challenge that any digital goods company faces is related to chargebacks or dispute cases. A chargeback is a right exercised by a credit card holder, by which the holder contacts the card's issuing bank and asks it to reverse an earlier made transaction.

Such operations are excessively costly for companies, especially for the early stage companies, such as startups. Besides the actual payment that must be refunded, the companies are also charged with the bank's processing fees, which may sometimes exceed the price of the actual product. In case of Miils, this would be a rather common scenario, because the price of a recipe is usually way below the costs of the related fees incurred in case of the chargeback.

There exist no exact mechanisms which would prevent all of the chargeback situations, but there exist steps which could minimize their occurrence. One of them is providing a clear refunding mechanism for the product or service, so that the user would not have to resort to the bank to get the money back.

At the moment Miils does not allow any further purchases for the users who have issued a chargeback, until the issue is resolved. Webhooks are utilized for achieving these purposes.

5.6.3 Webhooks

Each of the webhook messages received by the Miils back-end is authenticated, to ensure the validity of its origin, i.e. the Stripe servers. In fact, the only piece of information extracted from a delivered webhook message is its ID number. The rest of the payload is discarded. After a webhook message is received, the Miils back-end queries for the rest of the payload information from the Stripe servers on its own. To prevent replay attacks, all received webhook messages are saved to the database. Subsequently, each time a new event is received by the back-end, it is verified against the already received ones. If the message has not been received previously, its payload will be

processed next. In the example of an opened dispute case, the back-end receives immediately a webhook message about it with related details. The message is processed and all the necessary data is saved in the database. Additionally, the user who has initiated the dispute case is blocked until the issue is resolved. Therefore, the user that has opened a dispute case is not able to purchase any new items on our service. This prevents the possibility of an exploit, where users open repetitive dispute cases to achieve the denial of service.

One of the challenges that we have faced while working on the webhook handling was related to their testing on our local development environment. Stripe servers send messages about any occurred events to nodes with public IP addresses only, as expected. Since a local environment is hidden behind a firewall and usually has a private IP address, it becomes unreachable from outside. To solve this problem and test the correct functioning of webhooks, we have utilized the ngrok tool [42]. The tool provides a public IP address for nodes behind the firewall. This public IP address is then specified in the Stripe dashboard, so that Stripe knows where to deliver the webhook messages. Subsequently, ngrok routes messages originated from Stripe to our localhost.

5.6.4 Terms of service

The terms of service (ToS) is another important topic pertinent to the area of digital services. In fact, this subject represents one of the most frequent legal matters encountered by startups, according to the study [3]. The terms of service topic carries significance, because if stated inappropriately it opens risks for liability issues, which may result for the company in large losses and even destruction.

With the higher number of parties participants in the payment process, the complexity of the terms of service increases. More exactly, for each relationship between two parties there must be defined a ToS applicable to this interaction. As per 3.2, there exists four sides: the marketplace, the payment provider, the merchants and the customers. According to the diagram, there are defined four direct relations. Therefore, there must exist four different terms of service agreements. Two of these relations are concerned with the Stripe services. As a result, the ToS for these two relations are defined by Stripe itself. Such, the Miils marketplace agrees to the Stripe's terms of service when it establishes an account and declares that it will be a marketplace. Merchants, on the other hand, agree to the Stripe's merchant terms of service when they sign up as merchants on the Miils website, which redirects them to the Stripe's website where they must to agree to the corresponding

terms.

The Miils customers agree to the service's terms of service whenever they sign up to our service. In the terms of service applicable to the customers it is mentioned that some of the private details (email address, name) will be shared with Stripe in order to complete the payment process. Likewise, merchants agree to their ToS when they become merchants on the Miils marketplace. In the agreement it is stated there that Miils will be acting on their behalf and will perform all the necessary operations. In addition, this particular ToS specifies the charges imposed by the marketplace services.

5.6.5 Technical details of Stripe integration

Lack of a Django app for Stripe Connect

The Miils application, which represents the subject of the case study presented in Chapter 5, chose Stripe payments as its payment provider.

Stripe Connect is a service provided by Stripe for building marketplaces. It enables web services to act as a middleware between merchants and customers. Merchants, registered with Stripe Connect, are the individuals from whom a customer purchases products. In case of Stripe Connect Standalone Accounts, merchants reserve responsibility for the quality of their product and, as a result, for any possible disputes, as mentioned in the previous sections of this work. This removes the burden of paying any charges concerned with dispute cases. Moreover, in case of any discussion or conflict, the customer can contact the merchant directly. In fact, merchant's name is displayed together with the rest of the details when the customer pays for a product. Of course, the marketplace represents the party facilitating the commercial operations and it participates in the discussions when required. However, in the majority of cases, its intervention is unnecessary.

Stripe provides a mechanism aimed at verifying the merchant's integrity. In other words, the marketplace does not have to build an additional authentication and verification mechanism for its merchants. Neither does the marketplace have to or need to store details about payments from its customers and merchants. Although the marketplace does not represent a merchant in itself, it is able to access the merchant's data. Due to that, the marketplace is able to moderate the process of sales.

Even though there exist a number of widely known services that use Stripe Connect, the service is still relatively new. Hence, the lack of packages for the web frameworks, such as Django. However, there exists a package for another service provided by Stripe, called Django Stripe Payments. It mainly focuses on subscription based services. The package was used in production by a

number of companies. However, at the time of the technical implementation for this work, it was not up-to-date with the latest Stripe library version. Therefore, it was partly outdated. Due to lack of an existing package for Stripe Connect, all the necessary functionalities were written by the thesis author.

Fortunately, Stripe provides a Python library for its API calls, which was utilized for the payment system integration into the Miils application. In addition, the author used the source code of the previously mentioned Django package as a reference during the integration phases.

Changes to components brought by integration

Miils has a set of defined APIs, which are open to the public. This will change with time, as Miils will introduce an authentication mechanism for potential partners (fitness applications, medical institutions) and beneficiaries of generated data. The partners and others will authenticate with the API keys, which will be generated individually for each party.

Miils marketplace uses Tastypie API framework for creating the REST-style interfaces. These are used internally by the system to modify the database data via AJAX calls. AJAX improves user experience, as it minimizes the number of redirects that user experiences on the Miils pages.

Currently there are nearly 20 open interfaces which are accessible to session of authenticated or anonymous users. A few of the interfaces must be open, because they provide essential service functionality. For instance, recipe browsing by an anonymous requires calling specific APIs, which cannot be accessed unless they are opened to the public.

With the introduction of payments, a new authorization rule was defined for the case of items posted for sale. Previously, anyone was able to read the details about any publicly shared item. However, the items created for sale should display their content only partially. On the other hand, a recipe published freely displays its content fully to anyone, even to anonymous users. A recipe posted for purchase has its nutrition estimates open to everyone, as well as the data about the allergies, but does not display either its ingredients or its preparation instructions. These details become available once the recipe is purchased. All of the recipe posted for purchasing, are public, as it may be expected. However, the "filters" applied to it differ from those applied to a regular recipe.

As a result of the payment system integration, APIs were modified to become more restrictive, so that they returned only limited data about purchasable resources. However, when an already purchased item is displayed, a different API is being called, which returns extensive details about the item.

Additions to the existing codebase

In this section we list the functionalities that have been added into the existing Miils codebase for payments integration.

Following functionalities have been added as functions to Django views.py, and as methods to models.py:

- Handling of Stripe Connect webhooks. The webhook messages received from the remote servers are parsed and all events are logged into the database. The function also verifies that no duplicate events are processed twice, by checking the ID which is also encapsulated in the messages. The events that have been captured are then validated: back-end servers send a request to Stripe servers together with the ID received to make sure that such event has been issued. Validated messages are further processed and handled accordingly. For example, in case a merchant has deauthorized the marketplace from its selling right, the corresponding merchant account is deleted from the list of selling users. Further, merchant's products are removed from the search.

- Authorization of the marketplace for selling on merchant's behalf. Details of this process are examined in section 5.5.

- Deauthorization of the marketplace from selling on merchant's behalf. The process of deauthorization is displayed in further detail in Figure 5.5. Thus, users selling products on Miils that decided to stop this activity can inform Miils directly on its web pages, without navigating to Stripe dashboard. When a merchant initiates the deauthorization process, a POST request is sent to a particular Stripe endpoint. The back-end receives then a reply stating whether the initiated action was successful or not.

- Changing credit card details. In case there exists no credit card information for a particular user, we save it for future use (however, neither credit card numbers, nor other sensitive data is ever saved in the Miils back-end – the rules that companies must follow to comply with PCI DSS A-EP category). Otherwise, the card details are updated and the new data is synced with the Stripe servers.

- Charging customers for a purchased item. The function performing this action creates a token for the charge (further details about tokens are provided in Chapter 5.5), which is subsequently submitted to Stripe to record the charge. Charges are also logged into Miils database.

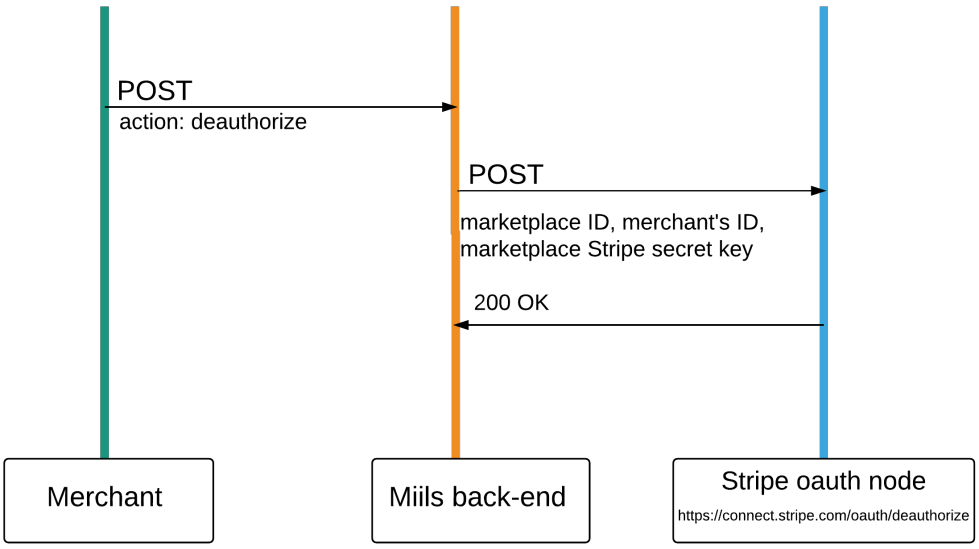

Figure 5.5: Deauthorization process: merchants deauthorize marketplace from selling their products

To accommodate the functionalities required by payments system, the following classes have been added to models.py:

- StripeObject: a generic class for all payments-related objects (charges, refunds). The model has two fields: Stripe ID (each event generated by Stripe has an ID) and the date and time of the event creation. All newly created classes inherit from this class.

- StripeSharedCustomer: a class of all customers on Miils marketplace. Customers are users who have purchased at least once. For each of the customers, Miils stores particular details about their credit cards (four last digits of the credit card number and card's expiry date). Additionally, each customer object has an attribute called "blocked". This attribute serves as a tag, reflecting the current status of this user. In case, a user violated the Miils terms of service, or has opened a dispute case against a merchant, this user is tagged as blocked. Such users are not able to purchase any items on Miils, until the flag has been removed.

- ConnectEvent: a class for Stripe events. It includes the above mentioned validation and processing methods.

- StripeConnectDisputes: a class for dispute cases, which thoroughly documents the reasons, the status and the parties involved in the dispute.
- StripeCharge: a class for purchases, which includes details about the charge.
- StripeAccessTokens: a class of tokens issued to the marketplace on the moment of authorization. One of the attributes of this class objects is the merchant's publishable key - a public key assigned by Stripe to each merchant. In fact, this key is what allows the marketplace to manage payments on merchant's behalf.
- EventProcessingException: a class of all exceptions that occur during payment processing.

Stripe Connect versus Transfers API

Miils payment system's design and implementation has significantly changed during its development phases. At first, Miils aimed at receiving all of the payments from the customers to its own bank account and only afterwards transferring the funds to the corresponding merchants. However, this solution proved to be limiting and burdening from a few points of view:

1. Any dispute cases opened against any merchant will have to be settled by Miils itself. Hence, in case of a high dispute rate, Miils could quickly go out of business.
2. Miils would have to perform tax paperwork for all of its merchants (e.g., 1099-K form in the United States [30]).
3. Miils would be accepting only U.S. based merchants, according to Stripe's terms of service. That is due to the limitation of Stripe's Transfers API service.

Moreover, the Stripe Transfers API service has been deprecated, as of today. Instead of Transfers API, marketplaces can choose another Stripe solution - Stripe Connect with Managed Accounts. Miils chose, however, to implement the Stripe Connect Standalone Accounts. Beside the above mentioned reasons, Stripe Connect Managed Accounts allows only U.S. or Canada based marketplaces.

The details presented below specify the mechanics of a Stripe Connect marketplace:

1. All of the marketplace merchants acquire a Stripe merchant's account. Together with the Stripe's account, merchants gain the benefits of automated reports, email notifications, customized email receipts, and so on. They are also able to access the data related to their sales directly from their Stripe account's dashboard.

2. Each of the Stripe Connect merchants represents an independent seller. Therefore, they will settle any dispute cases on their own. This fact is crucial for Miils as a business. This may appear as a disadvantage to Miils merchants, since they will be the ones responsible for paying all of the settlement fees in case of a dispute. On the other hand, the merchants are the ones ultimately responsible for their products and their quality.

Besides its significant advantages, Stripe Connect has disadvantages too. One of them is related to the process of the Miils app authorization by its merchants. This may be a hurdle because not every individual understands the reasons behind it and its mechanisms.

Updates and maintenance

As with any other current software, payment APIs provided by the payment system providers are updated on a regular basis. Consequently, marketplaces must react, update and adapt its services accordingly. Payment providers do not force, however, the immediate update installation, in case these are backward incompatible. Instead, the marketplaces are informed of such updates and are asked to upgrade as soon as possible. Backward-compatible updates are rolled out automatically and these do not break any existing functionalities. In addition, Stripe provides the option of a roll-back to the previous API version in case service functionalities become unusable, and companies require additional time for fixing incompatibilities.

Chapter 6

Discussion

This work focused on payment solutions and their integration into existing web applications. Together with the new wave of emerging players in the FinTech market, there has appeared a modern generation of lucrative solutions that allow businesses to start selling their products online on the day of their registration. Owing to a high number of offers in this domain and a lack of clearly defined guidelines on how to choose a payment solution, businesses face the challenge of selecting a reliable and suitable service for their case. This challenge also becomes noticeable due to the lack of a pertinent literature on the subject – the domain of latest payment solutions is emerging and is constantly changing and has, thus, not been investigated thoroughly.

This thesis aimed at presenting the experiences of a company that chose to integrate a payment solution provided by the new generation payment service providers. Considering the scarcity of academic and practical articles on the subject, the research had to resort occasionally on a trial-and-error approach in all of the phases of product development. This work will, hopefully, benefit others who will encounter a similar situation. While this thesis does not present an exhaustive list of all conceivable situations in the integration of payments, it presents a perspective that the reader may consider valuable when implementing a payment system.

The field of payments represents a wide subject as it includes the obvious financial aspects, also legal, technical and a few other areas. This thesis attempted to introduce the majority of these pertinent aspects in order to demonstrate their relevance and how they affect the implementation and operations of a payment solution. For example, technically, any company will face the challenge of defining or modifying an already existing terms of service when introducing a payments system into their service. This task raises non-trivial questions that must be most likely reviewed with a specialized lawyer. Another challenge that any company faces after the implementation phase

of payment system is related to its update and maintenance—particularly, how to sustain a system operational and at the same time up-to-date.

The target audience for this thesis consists of small- to medium-sized companies and startups. To support this initiative, a case study is presented in Chapter 5, which describes the real-life implementation challenges faced by a small startup.

It is worthwhile noting that the topics regarded in this thesis may become obsolete with time due to the rapid development of the field of payment solutions. Together with these changes, simultaneously will advance the Miils service, described in the case study. This work, however, may serve as a reference point for related projects, as some of the aspects depicted here will perhaps remain relevant in the future as well.

It can be argued that among the limitations of this work is its somewhat narrow focus. For example, this research concentrates mostly on payment solutions for small businesses who require a quick solution and rapid deployment, which is not necessarily everyone's preferred choice. The situations illustrated in this work perhaps will not attract the attention of larger and established companies. In addition, the U.S. market is generally considered, although the thesis reflects on some of the European market solutions as well. Furthermore, the decisions undertaken during the selection and implementation phases were realized either individually by the author of this thesis or in best cases by a small group of people. Hence, there remains an open question about the subjectivity of this work.

Although the subject of the thesis is rather concise, the areas that considered in this research belong to distinct fields (such as business, technology, and law). Therefore, the text of this thesis represents to some extent a comprehensive manual on how to implement a modern payment system in a web service.

Future suggested steps for this work consist in observing the future developments of the described payment solutions and updating this research with relevant details and improvements. Concurrently with the development of the Miils service, other previously disregarded vantage points will uncover, which represent another valuable source of research in this area. Additional challenges encountered on the development path and their possible solutions will complement the current work with further depth and new perspectives on the domain of payment solutions integration for small businesses. Furthermore, the Miils web service might have increased its customer base by the time this work is reviewed again in the future. This opens a whole new area of investigation—how to upgrade a payment system supporting a small business to support a larger one, and what are the required changes for achieving this task.

Chapter 7

Conclusions

Electronic commerce is constantly evolving, and these changes are driven both by the technology as well as the human behavior. Along with the growing demand for the online interface of offered services, payment systems are developing as well, which enable web and mobile applications with their payment capabilities. The latest developments in this area provide a simple and convenient solution for integrating payment APIs into existing products.

When choosing a payment provider, companies must consider a number of aspects. First of all, from the business perspective they must review existing vendors and carefully research whether these suit their own business requirements, e.g., markets and payment currencies, settlement policies, dispute handling.

On the technology side, it is pertinent to review the integration procedures and available documentation. In the case of web development frameworks, it is worthwhile verifying that there exist readily accessible solutions to achieve fast and relatively effortless integration.

Merchants, especially small entrepreneurs, do not choose always to implement their own online store, as the task requires technical knowledge and considerable financial investments. They resort in these cases to existing marketplaces, which provide a fully featured platform for advertising their products. Naturally, marketplaces charge a fee for their services. However, marketplaces also provide a significant advantage from a number of perspectives for its customers—trustworthiness of products is often associated with the reputation of the marketplace.

This work comprises of several aspects related to the state-of-the-art payment solutions for small-to-medium sized businesses. Firstly, the work reflected on the most significant sides that must be considered when choosing a payment solution from the business and technical points of view. Secondly, the work contemplates on the integration of such services into existing on-

line services—the challenges and their possible solutions. Lastly, the thesis introduces a real-life case study of how a small startup has implemented a marketplace functionality to demonstrate the practicality and validity of points presented in thesis chapters.

Bibliography

[1] Accenture. Fintech Investment in U.S. Nearly Tripled in 2014, According to Report by Accenture and Partnership Fund for New York City, 2015.

[2] Angel, J. J., and McCabe, D. The Ethics of Payments: Paper, Plastic, or Bitcoin? *Journal of Business Ethics 132*, 3 (2014), 603–611.

[3] Armitage, A., Frondorf, E., Williams, C., and Feldman, R. Startups and Unmet Legal Needs. *Utah Law Review, Forthcoming* (2015).

[4] Auinger, A., Nedbal, D., and Wöss, W. A Holistic Approach for B2B Integration at Different Conceptual Levels. In *Proceedings of the 2009 International Conference on e-Learning, e-Business, Enterprise Information Systems, and e- Government (EEE'09)* (2009).

[5] Bureau, U. C. *Estimated Quarterly U.S. Retail Sales: Total and E-commerce*. US Government Printing Office, 2016.

[6] INFOGRAPHIC: The State of US Small Businesses. `http://www.businessinsider.com/infographic-the-state-of-us-small-businesses-2013-9`.

[7] Bussler, C. B2B Integration. In *B2B Integration*. Springer-Verlag Berlin Heidelberg, 2003.

[8] Capgemini. Top 10 Trends in Payments in 2016, 2016.

[9] Cretu, L.-G. Smart cities design using event-driven paradigm and semantic web. *Informatica Economica 16*, 4 (2012), 57.

[10] Delaney, L. J. Getting Paid. In *Exporting Essentials*. Apress, 2014, pp. 163–194.

[11] Dominic Broom, Head of Treasury Services EMEA, B. M. Innovation in Payments, 2015.

[12] Dwolla. https://www.dwolla.com/.

[13] Ecommerce Statistics Compendium 2010. http://econsultancy.com/us/reports/e-commerce-statistics/downloads/2076-econsultancyecommerce-statistics-uk-sample-pdf.

[14] Europe, E. European B2C E-commerce Report 2015. *Retrieved March 27* (2016).

[15] Payment Statistics for 2014. https://www.ecb.europa.eu/press/pdf/pis/pis2014.en.pdf?d9feb0268bb38b960b806ea69b6bf467.

[16] Key Facts on the new EU Consumer Rights Directive. http://ec.europa.eu/justice/consumer-marketing/files/crd_arc2014_factsheet-consumer_general_en.pdf.

[17] Facebook. Community Payments Terms. https://www.facebook.com/payments_terms.

[18] Forbes. What Is A Startup? http://www.forbes.com/sites/natalierobehmed/2013/12/16/what-is-a-startup/.

[19] Forum, W. E. The Future of Financial Services, 2015. Final Report from June 2015.

[20] Forum, W. E. The Future of Fintech: A Paradigm Shift in Small Business Finance, 2015. Federal Reserve Banks of New York, Atlanta, Cleveland, and Philadelphia, Joint Small Business Credit Survey Report 2014.

[21] Garrett, J. J., et al. Ajax: A new approach to web applications, 2005.

[22] Gatheredtable: Terms of Use. https://www.gatheredtable.com/terms.

[23] Returns and refunds on Google Play. https://support.google.com/googleplay/answer/2479637?rd=1.

[24] Gorton, I., Thurman, D., and Thomson, J. Next generation application integration: challenges and new approaches. In *Computer Software and Applications Conference, 2003. COMPSAC 2003. Proceedings. 27th Annual International* (Nov 2003), pp. 576–581.

[25] Guo, J., and Wong, C.-W. A Dynamic Account Payment Method for Integrating Heterogeneous B2C Electronic Payment Systems . *CIS Conference Papers* (2012).

[26] Hanjura, A. *Heroku Cloud Application Development.* Packt Publishing Ltd, 2014.

[27] He, W., and Xu, L. D. Integration of Distributed Enterprise Applications: A Survey. *IEEE Transactions on Industrial Informatics 10*, 1 (Feb 2014), 35–42.

[28] Huang, X., Dai, X., and Liang, W. BulaPay: a novel web service based third-party payment system for e-commerce. *Electronic Commerce Research 14*, 4 (2014), 611–633.

[29] The OAuth 2.0 Authorization Framework. `https://tools.ietf.org/html/rfc6749#section-1`.

[30] IRS. Form 1099-K, Payment Card and Third Party Network Transactions. `https://www.irs.gov/uac/Form-1099-K,-Merchant-Card-and-Third-Party-Network-Payments`.

[31] Jean-Charles Rochet and Jean Tirole. Must-take Cards: Merchant Discounts And Avoided Costs, June 2011.

[32] Kanhekar, H. R., and Mane, M. S. N. Digital Wallet. *Journal of Microcontroller Engineering and Applications* (2015).

[33] Kungpisdan, S. Modelling, design, and analysis of Secure Mobile Payment Systems, 2005.

[34] Leonard Richardson, S. RESTFul Web Services [M], 2007.

[35] Liebenau, J. M., Elaluf-Calderwood, S. M., and Bonina, C. M. Modularity and Network Integration: Emergent Business Models in Banking. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on* (Jan 2014), pp. 1183–1192.

[36] Lucchi, R., Millot, M., and Elfers, C. Resource oriented architecture and REST. *Assessment of impact and advantages on INSPIRE, Ispra: European Communities* (2008).

[37] Makai, M. Making Django Play Nice with Third Party Services, 2013.

[38] Manzoor, A. *E-commerce: an introduction.* Amir Manzoor, 2010.

[39] Michael Miller. The PayPal Official Insider Guide to Growing Your Business, 2011.

[40] Michelson, B. M. Event-driven architecture overview. *Patricia Seybold Group 2* (2006).

[41] Murgante, O. G. B., Taniar, A. L. D., and Gavrilova, Y. M. M. L. Computational Science and Its Applications–ICCSA 2008. *International Conference on Computational Science and Its Applications* (2008).

[42] ngrok. `https://ngrok.com/`.

[43] Octavian Ureche, Rejean Plamondon. Digital payment systems for Internet commerce: The state of the art, 2000.

[44] Paul Benjamin Lowry, Taylor Michael Wells, Gregory D Moody, Sean Humphreys, Degan Kettles. Online Payment Gateways Used To Facilitate e-commerce Transactions And Improve Risk management, January 2006.

[45] Paytrail. `http://www.paytrail.com/`.

[46] PCI SSC Data Security Standards Overview. `https://www.pcisecuritystandards.org/security_standards/`.

[47] Payment Card Industry (PCI) Data Security Standard Self-Assessment Questionnaire, April 2015. `https://www.pcisecuritystandards.org/documents/SAQ_InstrGuidelines_v3-1.pdf`.

[48] Plekhanova, J. Evaluating web development frameworks: Django, Ruby on Rails and CakePHP. *Institute for Business and Information Technology* (2009).

[49] Rauth. `https://github.com/litl/rauth`.

[50] Ruiz-Martínez, A. Towards a web payment framework: State-of-the-art and challenges. *Electronic Commerce Research and Applications 14*, 5 (2015), 345–350.

[51] Sahota, A. Payment Card Industry Data Security Standard. *IT Assurance and CAATS* (2014).

[52] Sheng, Q. Z., Qiao, X., Vasilakos, A. V., Szabo, C., Bourne, S., and Xu, X. Web services composition: A decade's overview. *Information Sciences 280* (2014), 218–238.

[53] Authentication with Connect. `https://stripe.com/docs/connect/authentication`.

[54] PCI DSS Self-Assessment Questionnaire (SAQ). `https://support.stripe.com/questions/do-i-need-to-be-pci-compliant-what-do-i-have-to-do`.

[55] Reference documents for Checkout.js. `https://stripe.com/docs/checkout`.

[56] Reference documents for Stripe.js. `https://stripe.com/docs/stripe.js`.

[57] Stripe. `https://stripe.com/fi`.

[58] What about PCI DSS 3.1? `https://support.stripe.com/questions/what-about-pci-dss-3-1`.

[59] What permissions do platforms get when I connect my Stripe account? `https://support.stripe.com/questions/what-permissions-do-platforms-get-when-i-connect-my-stripe-account`.

[60] Stripe Connect Platform Agreement. `https://stripe.com/connect/terms`.

[61] Turban, E., King, D., Lee, J. K., Liang, T.-P., and Turban, D. C. *Electronic commerce: A managerial and social networks perspective*. Springer, 2015.

[62] Weinberger, J., Saxena, P., Akhawe, D., Finifter, M., Shin, R., and Song, D. *Computer Security – ESORICS 2011: 16th European Symposium on Research in Computer Security, Leuven, Belgium, September 12-14,2011. Proceedings*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, ch. A Systematic Analysis of XSS Sanitization in Web Application Frameworks, pp. 150–171.

[63] Xu, J. *Managing Digital Enterprise: Ten Essential Topics*. Atlantis Press, Paris, 2014, ch. Digital Payment Systems, pp. 159–175.